

UNIVERSITY OF TARTU VILJANDI CULTURE ACADEMY
BALTIC FILM AND MEDIA SCHOOL OF TALLINN UNIVERSITY
Sound Engineering Arts

Stanislav Barvinskyi

SEQUENCER MODEL WITH VARIABLE DOMAIN BASED ON
MATHEMATICAL CONCEPTS AND ABSTRACT ART

Master's Thesis

Supervisor: Janar Paeglis, MA, Lecturer

Co-Supervisor: Hans-Gunter Lock, MA, Lecturer,
Estonian Academy of Music and Theatre

Viljandi 2014

This thesis conforms to the requirements for a Master's thesis

.....(signature of the supervisor and date)

Submitted for defense (date)

I have written this Master's thesis independently. Any ideas or data taken from other authors or other sources have been fully referenced.

I agree to publish my thesis on the DSpace at University of Tartu (digital archive) and on the webpage of the Centre for Baltic Studies, UT.

..... (signature of the author and date)

Table of Contents

INTRODUCTION.....	4
1. SEQUENCING THEORY	7
1.1 Mathematical terminology.....	8
1.1.1 Function	8
1.1.2 Sequence	9
1.1.3 Vectors and matrices.....	9
1.2 Spectra and vertical structures	10
1.3 Horizontal structures.....	12
1.4 Musical sequencers and their main types.....	13
1.5 Amplitude envelope definition and main typology.....	17
1.6 Oscillation and filtering.....	20
2. ABSTRACT GEOMETRY IN ART	22
3. COMPUTATION.....	25
3.1 The Sequencer overview	25
3.1.1 Defined sequences and patterns.....	26
3.1.2 Sequence scaling	28
3.1.3 The clock system	29
3.1.4 Amplitude envelopes and their main parameters	30
3.1.5 Data routing inside the envelope row.....	34
3.1.6 Matrix sequencer block.....	36
3.1.7 Oscillators.....	37
3.2 Formative idea behind the model	38
3.3 Testing.....	39
CONCLUSION.....	42
REFERENCES.....	44
SUMMARY	47

Table of Figures

Figure 1.1 Function	9
Figure 1.2 Horizontal structure with durational notes.....	13
Figure 1.3 Complex envelope output with series of four Attack-Decay (AD) envelopes and threshold as a trigger unit.....	15
Figure 1.4 Software sequencer of the Digital Performer,	17
Figure 1.5 Classic linear ADSR envelope curve.....	19
Figure 1.6 Frequency modulation domain.....	20
Figure 2.1 Composition With Red, Black,Blue, Yellow And Grey by Piet Mondrian	23
Figure 3.1 Model scheme	26
Figure 3.2 The scaling block.....	29
Figure 3.3 Example of the clock system	30
Figure 3.4 Amplitude envelope	32
Figure 3.5 Data routing inside the envelope row.....	35
Figure 3.6 Data routing sub-patch for the dictionary "main".....	36
Figure 3.7 Data routing sub-patch for the dictionary "localPass".....	36
Figure 3.8 Matrix Sequencer with 4 by 16 matrix size,	37
Figure 3.9 Oscillator	38
Figure 3.10 Spectrum and waveform view of the created pattern.....	39
Figure 3.11 Line 1 test.....	40
Figure 3.12 Line 2 test.....	40
Figure 3.13 Line 3 test.....	41
Figure 3.14 Line 4 test.....	41

INTRODUCTION

In recent times, software and hardware devices have become more and more popular due to heightened interest in electronic and computer music. Both artists and technicians work on the design and development of various electronic instruments and devices. With the aim of optimizing workflow, most designers base their work on the principle "less is more", trying to reduce the number of functions whilst at the same time preserving a wide range of possible outputs. On top of this, rapidly evolving technology has shifted the role of the artist and merged it with the role of designer. This has resulted in the theoretical and practical fusion of science and art. The most progressive electronic music artists start their work not at the composition stage, but much earlier - the designing of the technical tools for their projects. Consequently, electronic music has become a conceptual bridge between music, art, architecture and science.

The aim of this project is to design and develop a sequencer model as an artistic template which will allow the artist to develop, modify and use his own abstract patterns in real time. The basic idea of this project comes from the investigation of possibilities of analogue sequencers, such as Doepfer A-142 Quad AD/LFO and A-143 Complex ADSR models. These processing blocks provides user with number of possibilities of pattern developing, yet they are not precise and quite inconvenient in use. The main objective of the instrument is to provide the artist with flexible workflow for artistic live performance and improvisation. The approach the author has used considers both mathematical and creative perspectives on the structure, equilibrium, aesthetics and "painterliness" of sound as both an individual object and as part of a pattern. The author implements a new approach to the development of sequence functions, in which the method used is the alteration of function domain.

The author believes that such a sequencer design will open up new possibilities for laptop live performance and move the aesthetics of electronic music towards better expressiveness. The

"less is more" principle is aimed at simplifying artistic workflow on the one hand, and optimizing it on the other. This is certainly the main technical and musical objective, as it provides the artist with a flexible template for real time performing which, in turn, does not limit the creativity of the artist.

In order to systemize the functionality of the designed sequencer and its supplementary blocks, as well as to conceptualize the artistic output, a range of fundamental mathematical and artistic concepts are investigated and implemented in the course of this project. The concepts of mathematical sequences, vector and matrix are essential because the sequencer is designed as a 4x16 dimensional matrix and its domain is controlled by the components of vectors. These vectors define time ratio between the steps in the sequence. The spectra and shape of every sound object will be created by the combination of complex FM oscillators and an amplitude envelopes chain.

In this regard, the structure of the work is as follows: each chapter consists of a brief explanation of the mathematical and music theory used for the model design, and provides comments on step-by-step practical realization. In so doing, the author hopes to make the work comprehensible to a wide readership and possible users.

Chapter one lays the foundation by introducing the basic mathematical concepts needed to comprehend the musical structures. In this paper essential mathematical terms such as sequence, pattern, function, vector and matrix are used to represent the principle and structure of the designed model. Additionally, the essence of musical vertical and horizontal structures are also explained. Vectors, sequences and their components aim to replace the traditional music notation determinants.

Chapter two draws attention to the specific artistic concept of abstract geometry in Neoplastic art. According to the author's viewpoint, the essential principles of abstract art together with the aforementioned mathematical concepts can be successfully implemented as an alternative to obsolete conservative music notation rules. In this regard, the chapter offers a brief introduction to the key principles of neoplasticism and elaborates on its close connection to electronic music. This approach is theoretically expanded and is not practically implemented in the model. However, the author plans to work on the practical realization of this concept in the future.

Chapter three elaborates the development process and computation of the designed model. It starts with a general overview of the device and its scheme. The computational process and data routing are explained in detail. The main control parameters of the matrix sequencer, amplitude envelope chain and other parts of the project are reviewed. The author also explains the fundamental idea of the project and elaborates on its artistic origin.

Chapter four contains a short artistic and technical estimation of the work and offers suggestions for the possible modification and optimization of the given instrument. It summarizes the results of the project and sheds light on the prospects for its future development. Additionally, the author analyses possible drawbacks and outlines how these could be eliminated.

It is important to mention that the author does not consider the designed device as an imitation of natural musical instruments, and therefore the possible output of the designed template lies in between electronic music and sound art. Correspondingly, the work does not aim to investigate the concepts of music composition, notation or musical aesthetics. Instead, the author proposes a new concept of performance and composition, which can be embodied with the new abstract sequencer model. This template is suitable for electronic music production, and considers the concepts of avant-garde visual art and the essence of mathematical sequences and patterns. Lastly, the designed model has to be considered as the first phase of the artistic project.

1. SEQUENCING THEORY

Almost all processes in nature are represented and understood through numbers and functions. Music notation with various scale technics, rhythmic patterns, hardware and software sequencing, sound synthesis, filtering and other processing - everything is functioning according to particular rules and theoretically explained by mathematical functions. Correspondingly, the structures of sound events in time or frequency domain also follow rules. Therefore the concepts of mathematical sequences and patterns become crucial for understanding how sound is organized in time and frequency domain.

1.1 Mathematical terminology

1.1.1 Function

A **function** from a set **D** to a set **Y** is a rule that assigns a unique element $f(x)$ to each element x in the set **D** (G.B.Thomas and Maurice D.W. 2004, p. 19). This process is depicted on Figure 1.1. It simply shows that if a is some element in a set **D**, it has some corresponding element $f(a)$ from a set **Y**, and the function $f(x)$ is a rule, which makes this assignation. In this case **D** is a **domain** of the function, in other words, it is a set of all possible inputs, and **Y** is a **range** of the function, and it is a set of all possible outputs (Figure 1.1). The value of output, commonly written as y or $f(x)$, is determined by value of the input (variable), usually written as x , and y in this case is defined as a function of x .

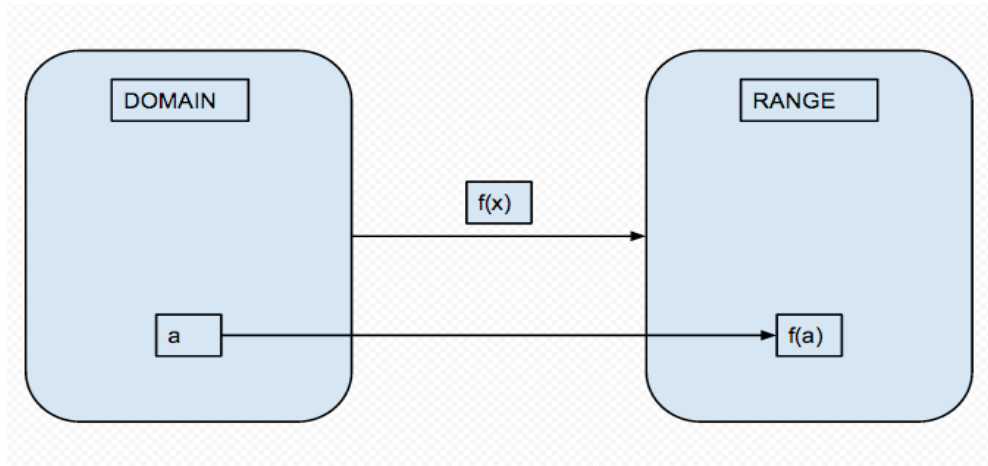


Figure 1.1 Function

1.1.2 Sequence

Sequence is a real-valued function with the domain of natural numbers (*Sequences and series* 2013, p. 13). Sequences are defined by particular rules and they depict a system behind the patterns. In other words, a sequence is a string of the objects (numbers, functions, geometrical objects, etc.) that follow a particular pattern.

The way how the objects are set in the list depends on the rule behind the sequence, which is called a **pattern**. Taking into account that theoretically the sequence's terms are natural numbers, the sequence then is denoted as a function with the domain of the natural numbers.

General notation of a sequences is following (*Ibid.*, p. 14):

$$a_1, a_2, a_3, \dots, a_n$$

$$(a_n)_{n=1}^{\infty}$$

$$(f[n])_{n \in \mathbb{N}}$$

Each output in a sequence list is called a **term**, where a_1 is initial term and n is a number of terms in sequence.

1.1.3 Vectors and matrices

In this work the author uses vectors as rules, which define the patterns for every channel of the sequencer. A vector notation is convenient because of basic properties of vectors.

A k -dimensional **vector** v is an ordered list of k real numbers $[v_1, v_2, \dots, v_k]$, which are called the components of the vector v (Bradley, Hax, and Magnanti 1977, p. 487). Within all vector spaces it is possible to multiply a vector $v = [a_1 a_2 \dots a_n]$ by scalar c (Strang G. 1991), and result will be a new vector in the same vector space R^n :

$$[c * a_1 \ c * a_2 \dots c * a_n]$$

A **matrix** defined as a list of lists, or rectangular array of numbers, whose dimension is m by n (Bradley, Hax, and Magnanti 1977, p. 488). A simple 2 by 2 matrix is written as:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

The numbers a_{ij} are the elements of the matrix.

1.2 Spectra and vertical structures

In music and sound synthesis everything from the timbre spectra to composition timing can be defined as sequences, which have patterns. Musical staff is mathematically interpreted as 2D vector space, with a horizontal x axis and a vertical y axis. Music events, such as notes - their length, rhythm elements and other episodes - are perceived through an interval of time, therefore time is usually performed by a horizontal axis x (Wright 2009, p.17). Vertical axis y or **vertical structure** assigns pitch on a music staff in notation domain.

Harmony of sound spectra is determined by frequency ratios in a frequency domain. According to Fourier theory (Smith 2002; Benson 2007; Puckette 2007; Prandonini and Vertelli 2008), every complex periodic waveform $X[n]$ can be decomposed as a sum of sinusoids with different frequencies and can be depicted as N -terms list of angular frequencies:

$$\omega, 2\omega, 3\omega, \dots (N-1)\omega,$$

where ω is the fundamental angular frequency and all following terms up to $(N-1)$ are the partials or overtones. This is a very common sequence of the harmonic sound partials, where all the terms are integer multiples of the fundamental frequency ω and it obviously follows the next pattern:

$$(n * \omega)_{n=1}^N$$

where ω is a fundamental frequency, n is an index of the term, $n=1$ is a initial term index and N is a last term index. Another essential pattern in timbre representation defines a phasor behavior. In a complex exponential form each sinusoid is defined by the formula (Smith 2007; Wickert 2011):

$$z(t) = Ae^{j(\omega_n t + \phi)} = Ae^{j\phi} * e^{j\omega_n t} = Xe^{j\omega_n t}, \quad (1.1)$$

where phasor X is complex amplitude A and phase ϕ applied to the time varying component of frequency $e^{j\omega_n t}$, e is Euler's constant j is an imaginary number. The equation (1.1) comes from Euler's formula for a complex sinusoid (Strang G. 1991, p. 362):

$$\cos(\omega + \phi) + j * \sin(\omega + \phi) = e^{j(\omega + \phi)}$$

By setting the rules for the phasor and frequency components, it possible to synthesize any probable complex waveform. Harmonic sound as a sum of positive and negative frequencies can be depicted as a set of frequency-amplitude pairs (Wickert 2011):

$$\{(0, X_0), (f_1, X_1/2), (-f_1, X_1^0/2), \dots (f_N, X_N/2), (-f_N, X_N^0/2)\}, \quad (1.2)$$

where $f_n = \omega_n / 2\pi$, $-f_n = -\omega_n / 2\pi$ and $X_n/2, X_n^0/2$ are the pairs of the positive and negative oscillation frequency and phasor of the n term respectively.

Presume the given initial condition is a pair (f_1, X_1) , simply put, a sine with frequency f_1 and arbitrary amplitude and phase encapsulated in the complex phasor. For a simplification of the process, the negative sides are not used. By applying a simple pattern to this

list it is possible to synthesize the classic waveforms. For instance, the result of multiplication of a given pair and a corresponding pair

$$\{ (1,1), (2,1/2), (3,1/3), \dots (n, 1/n) \}$$

is the sawtooth waveform.

Consequently, each partial of the sawtooth follows the rule:

$$(n * f_1, 1/n) ,$$

where n is the number of the partial. If the pair is multiplied by the list

$$\{ (1,1), (0,0), (3,1/9), (0,0), (5,1/25) \dots, (n, 1/n^2) \},$$

the result will be a symmetric triangle waveform. This is the basic principle of the additive synthesis (Miranda 2002, p.50).

Amplitude of every partial here is a scaler of the amplitude of the fundamental, and the frequency is integer multiple of the fundamental frequency. This example explains the spectra development in additive and wavetable synthesis.

It is very convenient to represent a spectrum in exponential form using the frequency domain. It clearly illustrates how mathematical rules can be applied for timbre spectrum representation and sound synthesis.

1.3 Horizontal structures

Horizontal structure - is a sequence of sound events and episodes in time domain (Wright 2009, p. 17).

In **horizontal structures**, as a time scale, sequences are conventionally defined by the beat, tempo, and duration of the notes (Figure 1.2). **Beats** are the time intervals "count-off" units, by which traditional music is notated (*ibid.*, p. 17). In electronic dance music, such as house or

techno, beats often refer to kick drum episodes. **Tempo** in this case is a frequency of the beats and it is defined in beats per minute. In pop, rock and dance music the tempo usually does not change during the track, so beat duration is also a constant. The **note** is the main duration determinant in a music score. Depending on length and tempo, durational notes are defined as whole note, half note, quarter note, eighth note, sixteenth note, thirty second note, sixty fourth note (*ibid.* p.18). For instance, if the whole note includes four beats (simple quadruple), then a half note gets two beats, quarter - one and so on. If the tempo of a song is defined as 120 beats per minute, then one whole note duration will be $60/(120/4)=2(sec)$.

Figure 1.2 Horizontal structure with durational notes (*Source*: Wright 2009, p.19)

$$d_m = d * (2 - (1/2)^m) \quad ,$$

Since tempo represents the frequency of the beats, it can be depicted as a list of the beat indexes: 1,2,3,4; 1,2,3,4;.. for a commonly used simple quadruple, where whole note equivalents - bars, are separated by semicolons. If the tempo is 5/4, then it can be depicted as 1,2,3,4,5; 1,2,3,4,5;.... The beats in this case are equal in length if tempo doesn't change. For musicians sometimes it is more convenient to think about musical sequence structure in terms of beats/bars and other division levels and episodes, which are determined by durational notes and rests. In software sequencers and editors the length of the cells depends on a selected quantization mode,

which is the actual domain of the sequencing function. This option helps a user to specify a precise time for every unit of horizontal structure.

1.4 Musical sequencers and their main types

Most music sequencers are based on repetition of the sequence, thus they are able to repeat the same events systematically. Because electronic music is generally associated with the technological progress, the technical innovations in designing such models have been playing a crucial role in the development of electronic music over the last 30 years. Many of electronic music genres were established due to the limitations of the instruments. Consequently, by analyzing such limitations or inconveniences we can comprehend the fundamental idea behind the horizontal structures of the various electronic music genres.

There are four main types of the sequencers on the market: analogue sequencers, step sequencers, MIDI sequencers and digital or DAW sequencers (Roads 1996). All these four types have a long history and have been intensively used by electronic music composers for around 30-40 years. All the four types evolve and successfully maintain popularity in certain areas of electronic music production and performance.

The step sequencer became extremely popular in the early 80th. This type of sequencing was used primarily for drum programming. A classic step sequencer, probably the simplest but still highly popular type of sequencer, is generally limited to 1/16th note representation. Drum machines, for instance TR-808 or TR-909, are equipped with step sequencer, which allows the user to repeat usually up to 16 equal steps or segments from the selected pattern. So that this classic step sequencer has a function domain $[1, 16]$.

Modular synthesizers are more flexible in terms of arranging sequencing. They are normally provided with the analogue sequencers, which allow the user to modulate the events in a non-linear way, in other words they give the user the possibility to produce irrational structures. Compared to the step sequencer type, analogue devices require much more effort and knowledge for managing the horizontal structure. Such sequencers can be configured either manually or through voltage control connections, but in many cases it is extremely difficult or simply impossible to reestablish a recently created pattern or sequence. Analogue sequencers, without doubt, are the most artistic and flexible for improvisations regarding horizontal structure

manipulations, yet they do not provide the user with options for the implementation of defined patterns. Very often programming of relatively long patterns with such an analogue sequencer becomes tedious and unpredictable. The example of the complex envelope output is depicted on Figure 1.3

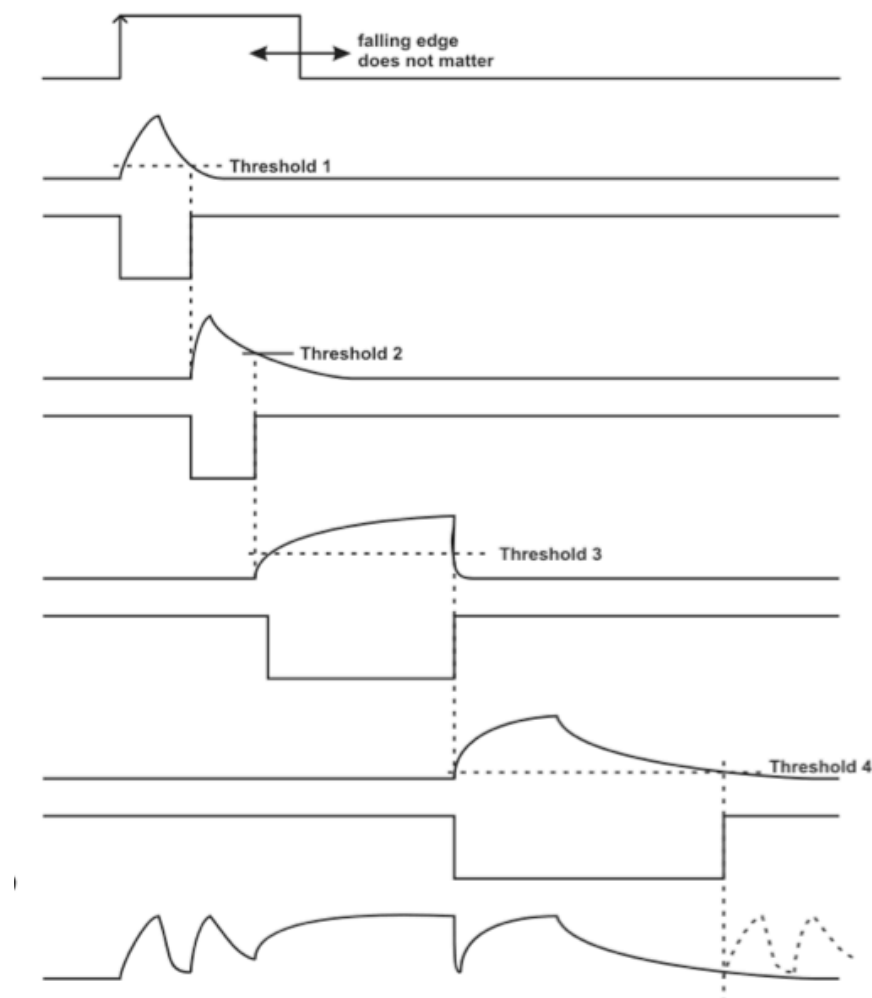


Figure 1.3 Complex envelope output with series of four Attack-Decay (AD) envelopes and threshold as a trigger unit. Top row is a trigger signal, next four rows are AD envelopes and the bottom row is a resulting complex envelope (*Source: Doepfer A-100 User's manual, 2009*).

In the digital domain sequencers are more complex; they are equipped with a number of various features, which are significantly helpful for the non-real time composing, recording audio and MIDI, editing and playback of recorded or composed material. These sequencers are implemented in the digital audio workstation model (DAW) and can be used for composing and

music production in non real time. Essentially, such sequencer operates the control data, such as velocity, note on, note off , various knobs, buttons, sliders and any other automation parameters. Every software sequencer in a digital workstation consists of arbitrary units for MIDI or audio data organization, and these units are conventionally called tracks (Roads 1996). There are four main types of tracks exist: MIDI track, audio track, aux track (in Ableton Live called Return) and master track. Some workstations, such as the Digital Performer additionally use an Instrumental track in order to separate instrument controls data from a MIDI note data.

MIDI information in MIDI channels can be represented in different ways (*ibid.*). The basic types of MIDI data representation are as follows: score editor, piano roll, drum roll and event list. The score editor is a feature that allows the user to write and edit a score using traditional music notation style with vertical and horizontal musical quantization, durational notes and corresponding rests for the horizontal, and note pitches for the vertical axis. The possibilities of input for score editors are strictly limited by musical notation. The example of software score editor is shown by Figure 1.4 (second track).

Some of these editors are the rule-based editors, and therefore they understand the basic rules of musical notation (*ibid*). They can automatically fix and control the pitch, duration of notes, provide the transposition operations, automatization and correction of the horizontal and vertical structures according to the musical notation rules.

The typical and most frequently used feature of the sequencer in DAW (Figure.) is a piano roll. The piano roll allows the user to write the score by filling the grid with appropriate notes. Vertical structure variables are the note pitches and horizontal structure depends on the selected quantization factor. Quantization factor defines minimum length of the note or rest that can be used (*ibid* ,1997). If a selected quantization factor is, for example, 1/16 note, than it is not possible to add the adjacent notes with duration less then 1/16.

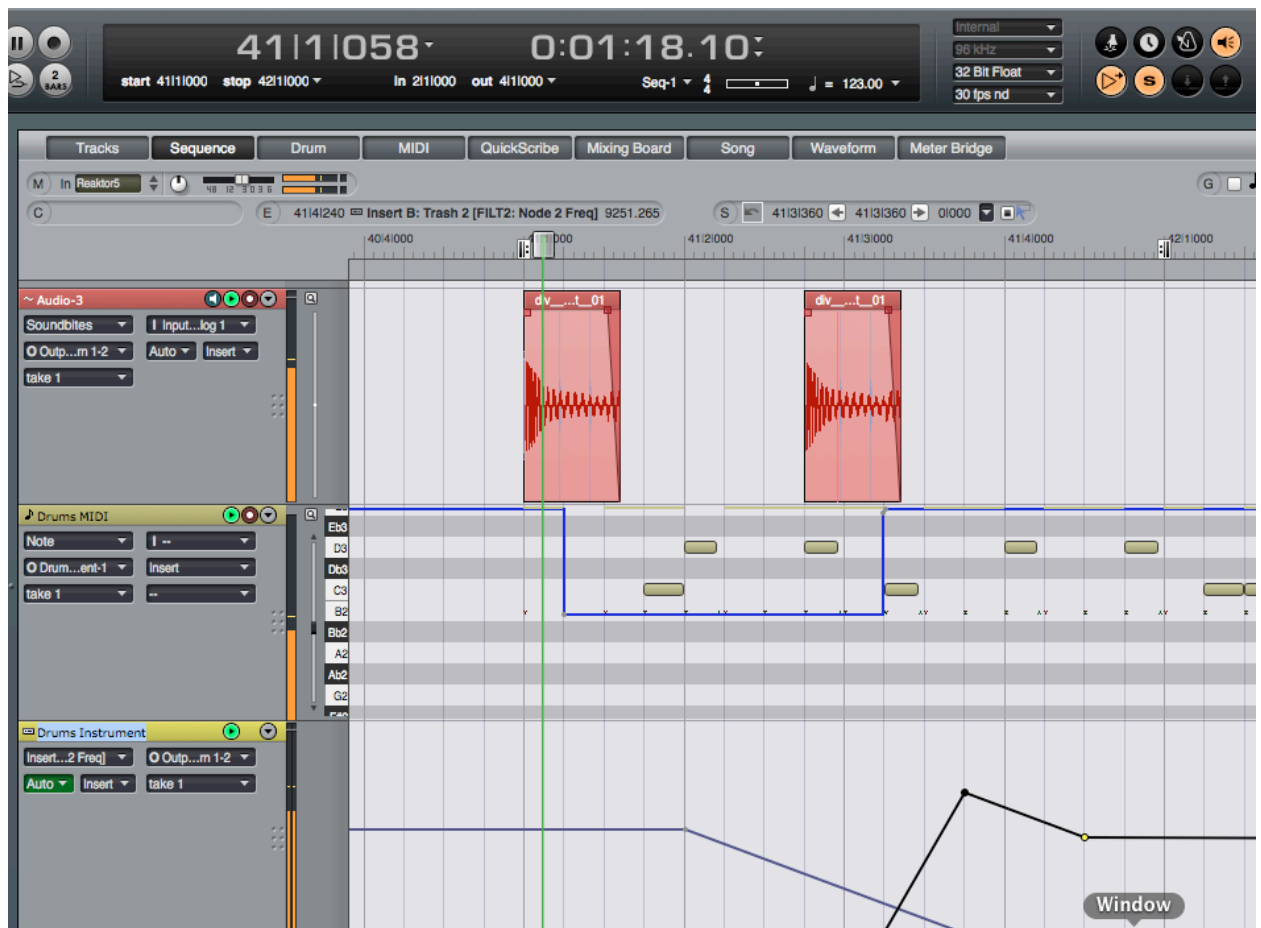


Figure 1.4 Software sequencer of the Digital Performer with audio track, midi track and instrument track

Although, most of software sequencers are endowed with a number of useful features, these systems are inconvenient for real time manipulations. Eventually, it may be inferred, that every type of sequencers has its own advantages and disadvantages regarding the fields of their possible use. Therefore one of the main objectives of the current work is to eliminate some of the most crucial disadvantages, taking into account the artistic goals of the work, and to compute the optimized hybrid model for the real time output and for realization of an artistic project.

1.5 Amplitude envelope definition and main typology

Horizontal and vertical structures are essential for computer music development, yet another significant moment should be considered additionally. The amplitudes of the musical units tend to change in time. For instance, percussive sound amplitudes have a short "fade in" and a bit longer, but still relatively short, "fade out". In computer and electronic music amplitude

envelopes are used for controlling these amplitude variations of the signal, produced by oscillation. Such envelopes control the amplitude changes of the sound objects in time.

The principle of amplitude envelope is based on signal multiplication, which in the discrete domain is a sample-by-sample, or "dot" multiplication (Ingle and Proakis 2012 , p.26):

$$\{x_1(n)\} * \{x_2(n)\} = \{x_1(n)x_2(n)\} \quad ,$$

where $x_1(n)$ and $x_2(n)$ are corresponding samples of multiplied signals. The expression from the right side of the equation gives the resulting value for a sample n .

An ADSR is one of the most common types of amplitude envelope generator and it is used broadly in synthesizers and samplers for a smooth rising, holding and falling (releasing) of the waveform amplitude. The letters ADSR stand for Attack-Decay-Sustain-Release. The amplitude envelope is a unipolar (or absolute) continuous function $f(x)$, where the arbitrary interior point $f(c)$ is an instantaneous amplitude of the signal produced by the oscillator at the time $x = c$. The domain of the envelope is set in milliseconds and its output defines the level of amplitude, which lies in the range $[0, 1]$.

The classic ADSR function can be depicted as a continuous piecewise function consisting of four segments: the attack segment (with positive slopes), the decay segment (with negative slopes), sustain (slop = 0) and release (with negative slopes):

$$f(x) = \begin{cases} g(x) & \text{if } 0 \leq x \leq x_a \\ u(x) & \text{if } x_a < x \leq x_d \\ w(x) & \text{if } x_d < x \leq x_s \\ v(x) & \text{if } x_s < x \leq x_r \end{cases} \quad ,$$

where $g(x)$, $u(x)$, $w(x)$ and $v(x)$ are functions of attack, decay, sustain and release respectively, x_a is attack time, $x_d = x_a + \text{decay time}$, $x_s = x_d + \text{sustain time}$ and $x_r = x_s + \text{release time}$.

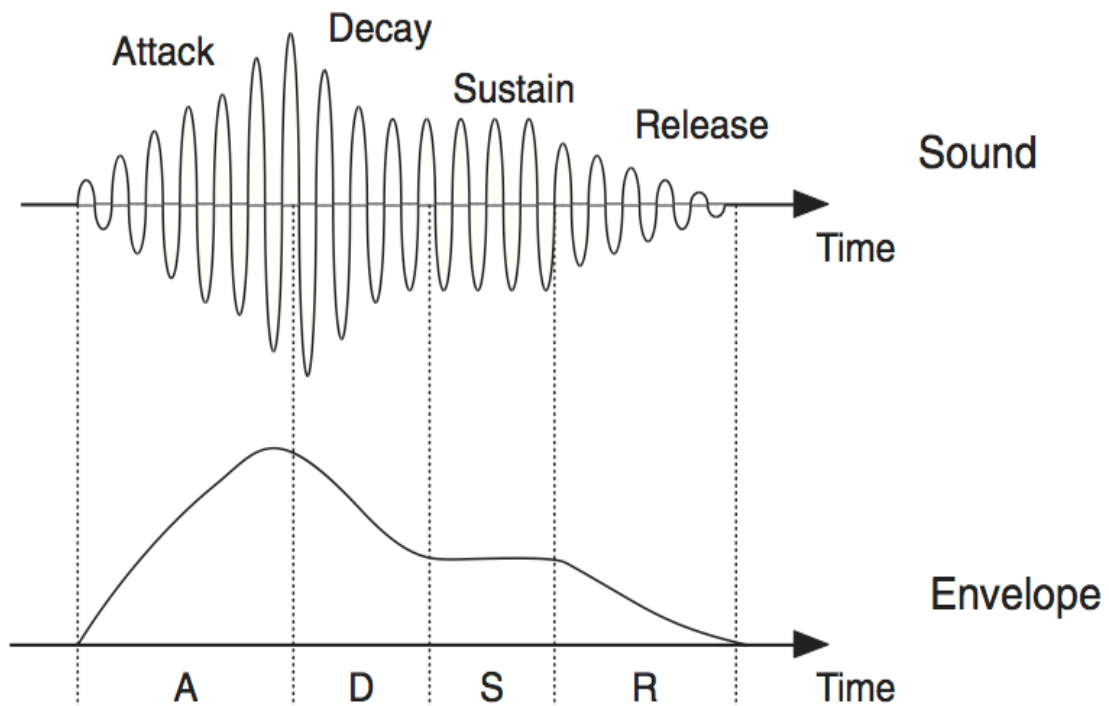


Figure 1.5 Classic linear ADSR envelope curve with four segments: attack, decay, sustain and release (*Source: Russ 2002, p. 44*)

In commercial MIDI synthesizers, amplitude envelopes are activated or triggered by pressing a key, what usually is called "note on". The maximum of the envelope's curve is determined by the velocity. Attack, decay and release are the functions of time. Sustain parameter in the keyboard-based synthesizer defines magnitude of the sustained note. In modular synthesizers, the beginning of the attack and the end of the sustain phase are normally defined by gate signal.

Unlike keyboard-based synthesizer or a real-time sequencer, artificial sequencers, like the software of hardware step ones, use a grid for the systematization of the triggering. This grid defines the domain of the sequence. Because of the limitations, caused by linking to grid, the compositions, made with sequencers, very often sound unnatural. The timing of human performance is never as mathematically precise as those produced by an artificial sequencers. In this case, such an "artificialness" comes out of the two simple limitations the typical sequencer has: **the function domain is limited**; in the natural world two successive notes are never the same in their amplitude shape. One can alter the envelope's parameters by using external controls

like LFO or manually, yet the level such a procedure becomes tedious with every new level.

Therefore, in many cases the commercial software sequencers have a set of limitations, and their sequencers and envelopes can hardly be of use for improvisation or comparatively complex patterns reproduction and management in real time. Irrational and intuitive patterns can be reproduced with keyboard-based instruments and modular systems where ADSR segments can be controlled manually or through voltage control. In this case, however, it is complicated to keep the structure logical or consecutive enough.

1.6 Oscillation and filtering

In order to demonstrate the possibilities of the designed model, the author uses four complex FM oscillators and one simple filter for every sequencer's channel. Frequency modulation (FM) synthesis was discovered by John Chowing in the late 1960s (Bilbao 2009 , p.6-7) and was implemented in commercial synthesizers by Yamaha in the early 1980s. **A classic simple FM model** (Figure 1.6) consists of carrier frequency, modulating frequency and frequency deviation, also called vibrato depth or modulation index (Chowing & Bristow 1986, p. 46).

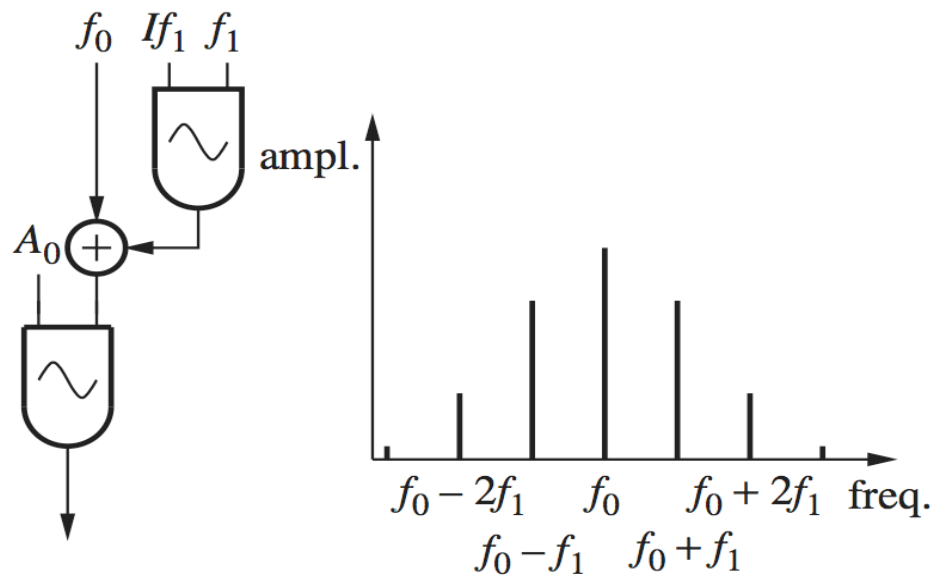


Figure 1.6 Frequency modulation domain (Source: Bilbao 2009, p. 7)

The principle of simple frequency modulation can be easily understood from the formula (Puckette 2006, Bilbao 2009):

$$x[n] = A_0 * \cos(I * \cos(2\pi * f_m * n) + 2\pi * f_c * n) ,$$

where f_1 is a frequency of modulation, f_0 is a frequency of carrier, n is a sample number and I is a frequency deviation factor or index of modulation, A_0 is amplitude of the output. With high values of modulation indices oscillators are able to produce timbres with very dense spectra. It does mean the output more likely should be filtered.

"The idea of a digital filter is to either reduce or increase the strength of a signal but only over specific frequency ranges (Leis 2011, p.271). The **digital filter** is a formula, which takes the sequence of samples as an input and produces the new sequence of samples, which is filtered output (Smith 2007). In this project the author uses simple two pole, two zero filter, which has four the most common options of filtering: lowpass, highpass, bandpass and bandreject (Cipriani and Giri 2011, pp. 300-307).

2. ABSTRACT GEOMETRY IN ART

The author's idea of implementation of mathematical patterns in the horizontal and vertical structures was inspired by the artistic philosophy of Neo-Plasticism – one of the most prominent art movements in the beginning of the XX century. The Neo-plastic art was developed in Holland in 1917 by De Stijl group, a community of the artists interested in pure form and geometrical style of the artworks. Escaping from the post-war reality, the art of the De Stijl group was meant to activate an inner vision capable to give its viewer the infinite prospectives on the Universe. As Gille Deleuze, a French philosopher, who gave an interesting and detailed comparative analysis of abstract art in his book on painter Francis Bacon, comments, abstraction is “a path that reduces the abyss or chaos” (Deleuze 2003, p. 103). “The abstract forms”, he continues, “are part of a new and purely optical space that no longer even needs to be subordinate to manual or tactile elements” (*Ibid*). Drawing further on Deleuze's givens, it becomes clear that the Neo-plastic art stands very close to mathematical patterns: abstract painting replaces a 'diagram' with a 'digital code', which consists of “units that group together visually the terms in opposition” (*Ibid*). As Deleuze puts it, the aim of abstract art is to “restore to man a pure and internal optical space, which will perhaps be made up exclusively of the horizontal and the vertical” (*Ibid*).

The proximity of Neo-Plasticism to mathematics was also highlighted by the famous painter, one of the prominent representatives of Neo-plastic art, Piet Mondrian: “Abstract-real painting can create in an aesthetic-mathematic way because it possesses an exact mathematical means of expression: colour brought to determination” (Mondrian 1919, p.286). By this, the painter connects artistic practice with scientific activity. Therefore, according to his beliefs, the artist is free to make any use of any science, any technique, and any material whatsoever to achieve preciseness (Beech 2009). The latter point is especially important, as it directly echoes the key aim of this work: to design an artistic musical template using precise mathematical patterns.

Inasmuch as a Neo-Plastic reality is represented by visual abstractions and forms, in electronic music reality is represented by abstract acoustic sequences. For instance, the structure of the painting of Piet Mondrian “Composition With Red, Black, Blue, Yellow And Grey” (Figure 2.1) could be interpreted as a geometrical pattern, in which the height of every rectangle can be explained as an amplitude of sound object; length – as the duration of sound object, and color determines intensity of the sound object timbre:

- grey – white noise;
- red – red noise;
- blue and black – low frequency region;
- yellow – high frequency region.

As a result, it is possible to read the painting two-directionally: either as the mathematical painting or as the musical sequence made of arbitrary rectangles.

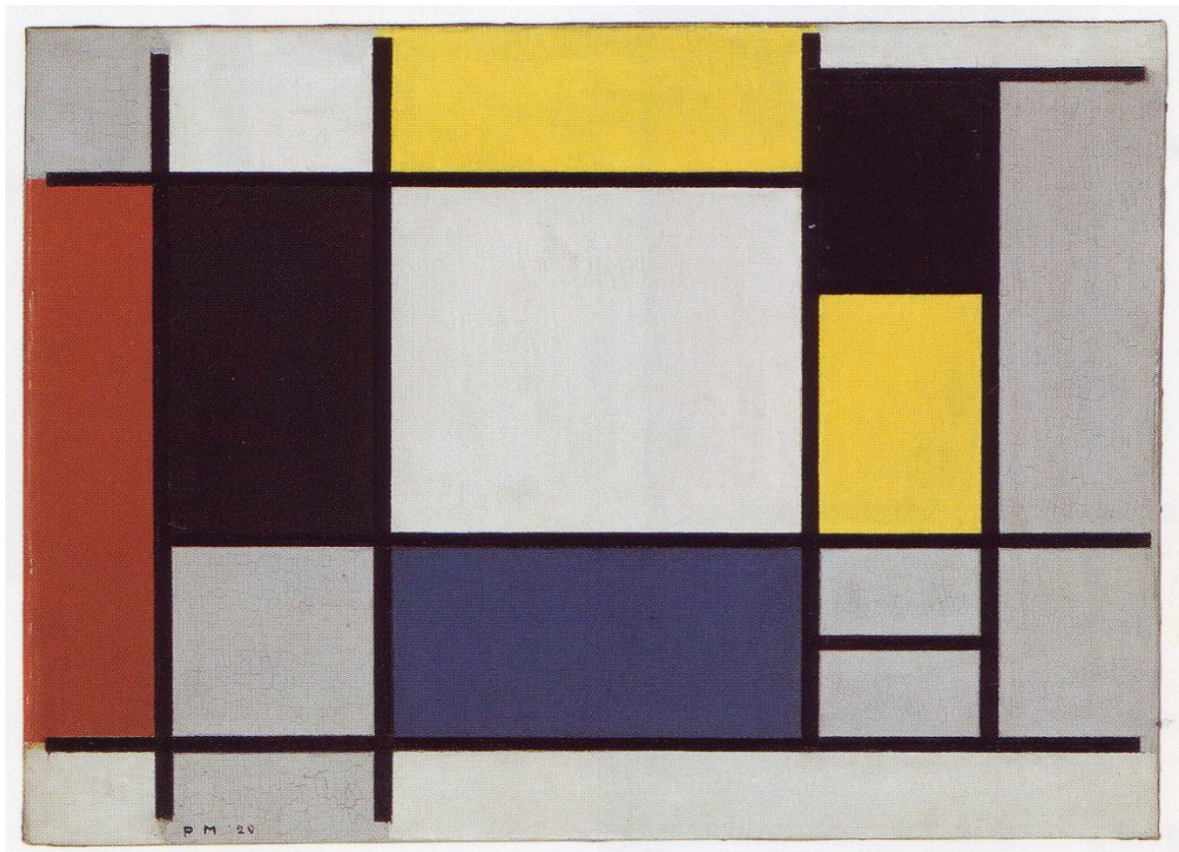


Figure 2.1 Composition With Red, Black, Blue, Yellow And Grey by Piet Mondrian

An additional principles used in relation to Neo-Plasticism in this work is the cancellation mechanism. According to an ideological leader of the Neo-plastic art Theo van Doesburg, cancelation is a forming parameter of the imaged equilibrium. Every element of the harmonious whole cancels out another: positions, dimensions, plains, lines and colors are canceled out by other positions, dimensions, plains, lines and colors respectively. Cancelation represents the essence of formative work, the fundamental idea of its aesthetics. (Doesburg 1919, p.283). In respect to the given project, it is precisely the cancelation mechanism which determines the structure of each sequencing line in the designing model. A more detailed application of the ideas of Neo-Plasticism will be described further in the practical part of the current work.

3. COMPUTATION

3.1 The Sequencer overview

The sequencer model, designed in this work, has an aim to reproduce rhythmical patterns defined by mathematical rules. The main accent here is made on a variability of the sequencer's domain, its possible non-linearity, and the amplitude envelope implementation for every sound object in a 16 step sequence. The sequencer is able to process 4 independent sequencing lines defined as row vectors. Each channel has its own pattern and amplitude amplification for every single event. Channels, if needed, can be synchronized in such a way that they could follow common pattern. Additionally, the sequencer can be modified by additive features like looping.

The model has complex data and the structure of signal processing, and it consists of the following blocks:

- a) pattern engine, where all rule vectors are defined;
- b) 4x16 matrix sequencer, where each row represents the line or the channel, and each column denotes the step index;
- c) clock system, which controls the tempo;
- d) set of independent amplitude envelopes for every cell;
- e) four independent FM oscillators;
- f) amplification blocks for each channel;
- g) series of two filters for every channel;
- h) pan control for every channel;
- i) master, recording block and tempo/step calculator.

The scheme of the model for one sequencing line is shown on figure 3.1

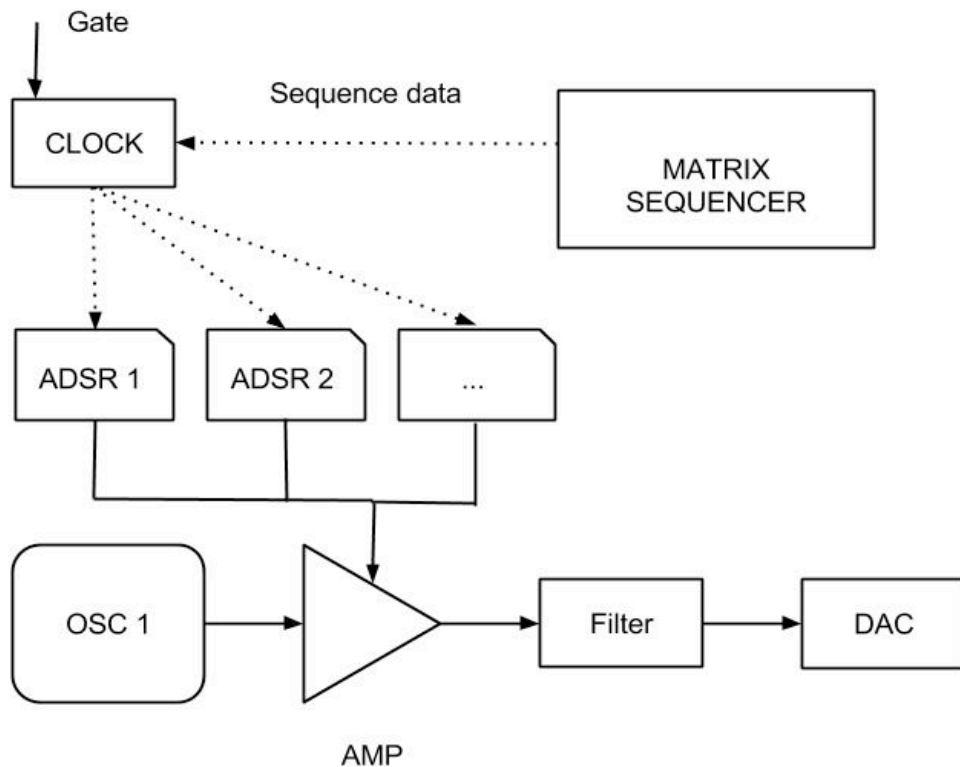


Figure 3.1 Model scheme

3.1.1 Defined sequences and patterns

In order to avoid linearity and periodicity on the one hand, and preserve aesthetics on the other, some mathematical functions need to be used. In this work author employs simple, previously defined, sequences and patterns for demonstration purpose. At the same time it is possible to add, store and call any desirable pattern.

The sequences for each line (channel) are set independently so that an artist can define the dimensions of the beat, note and tempo differently for every line and step. Horizontal structures in this case become mobile and irrational.

The sub-patch "expressions" contains and processes the data for a domain determination. The sequences, which can be used as a determinant of the domain, are called the **rule vectors**. They are stored in the objects "coll sequence-1", "coll sequence-2", "coll sequence-3" and "coll sequence-4" for every independent line respectively, and can be activated by sending a message

with appropriate index to the **coll** object's inlet. For instance, if an artist desires to define the rule for the second sequencing line with random numbers, he must choose "random" from the menu of line 2. The message with a number 5 will be sent to the "coll sequence-2" that will call the statement with an index 5, which is the list of random integers.

The sequences, which are stored as text and can be called from the coll objects, are as follows:

- Free. It is a vector of ones, therefore every cell gets the value of 1 as its length. This rule is convenient if a user wants to create a vector rule manually.
- Step. This mode is similar to the "free" mode, because it sends the list of numbers where all terms are equal, and the beat and tempo can be defined. Unlike the "free" mode, beat length here depends on the tempo, which can be selected in the main window exclusively for this mode.
- Sawtooth4. Sawtooth4 is a diverged and wrapped sequence with the 4 terms cycle:
1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4.
- Sawtooth3. It is a similar sequence to Sawtooth4, but it has a 3 terms cycle, and can be written as a list 1,2,3,1,2,3,1,2,3,1,2,3,1,2,3,1.
- Fibonacci. The Fibonacci sequence is one of the basic sequences in mathematics (Coxeter & Greitzer 1967, p.41). It can be expressed by a recursive formula $a_n = a_{n-1} + a_{n-2}$. For the first 16 terms then we have the following list:
1,1,2,3,5,8,13,21,35,55,89,144,233,377,610,987.
- Random. This sequence is a set of 16 terms, where every single term is an integer from [1, 9]. Each time random is called from the main patch, a sub-patcher named "random" generates a list of 16 arbitrary integers between 1 and 9 and writes this list as a statement with an index 5 to an appropriate coll object.
- Element-wise product of the Fibonacci sequence and its reversed version, named "Fibonacci contersequence". "Fibonacci contersequence" is an "element-wise" product of the original sequence and its reversed version. The operation takes the original sequence with its first 16 terms {1 2 3 4 5..} and multiplies it by reversed 16 terms sequence {987 610 377 233 144 ...} element-by-element (vector "element-wise" multiplication). The resulting sequence then will be: {987 1220 1131 932 720 ... }.

The last pattern demonstrates how a user can modulate the sequence domain by the

implementation of various mathematical patterns. The given model allows a user not only to employ previously defined patterns, but also modulate them in different ways by exercising simple mathematical operations. All mentioned sequences are stored as text and used for demonstrating the possibilities of the developed model.

3.1.2 Sequence scaling

Every cell (step) in the current model is defined in milliseconds. Each line, when it is defined by a rule vector, sends an appropriate value in *ms* for every cell to the clock system. The rule vector does not define the length of each step in a sequence, yet it does define the ratio of steps in one sequence. The vectors received from the "expressions" must be multiplied by scalar or some other vector (element-wise) in order to give a proper value of the length for each step in a sequence.

As a scaling factor this model uses scalars only, but it is possible to modify the sequence so that scaling factors will also be vectors or functions. Multiplying a vector by scalar is a basic vector operation, result of which is a new vector:

$$\{a_{s1} a_{s2} \dots a_{sn}\} = S * \{a_1 a_2 \dots a_n\}$$

, where vector from the right part of expression is a rule vector (row vector), S is a scalar multiplier (scaling factor), and the vector from the left is the actual sequence vector, where components are the length values of every step.

The scaling block of the given model is presented on Figure 3.2.

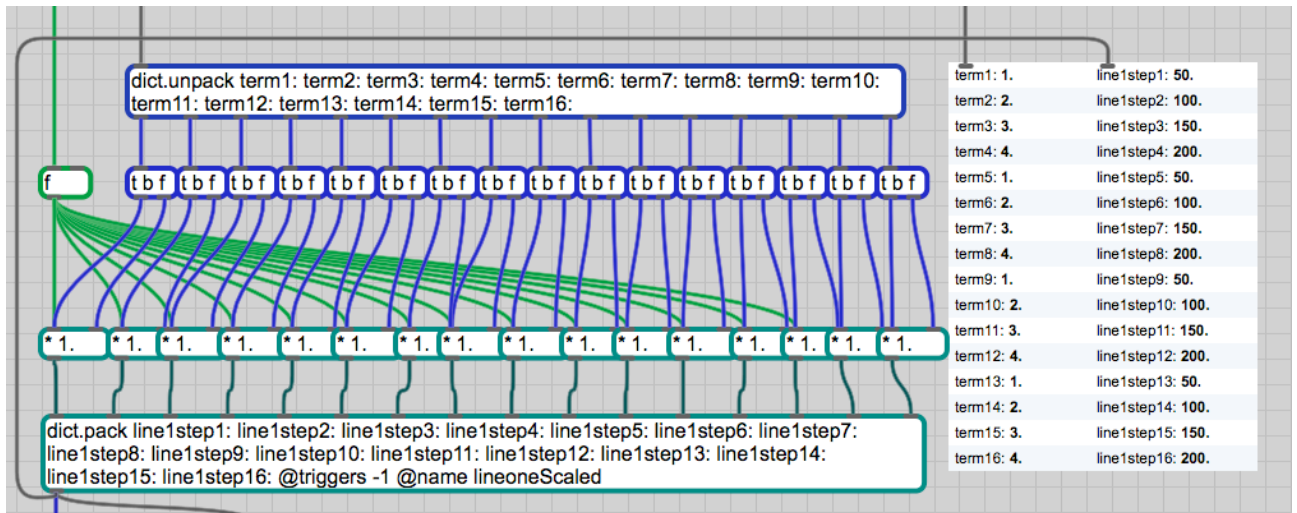


Figure 3.2 The scaling block. Here the dict.unpack object receives a vector with 16 components, denoted as term1, term2 and so on. Values of all components are shown in the left text column . Then these components are multiplied by a scalar (in this example scalar equals 50). The result thereby is a new vector with the values shown in the right text columns.

3.1.3 The clock system

The clock system manages the sequencer timing. Unlike a typical clock system for commercial sequencers, the system of this model is able to manage non-linear clock where instantaneous tempo for every step defined by the rule vector and its scaler. Simply put, tempo gets changed every time the clock system receives a length value for the new cell which is different from previous one.

The clock system is constructed so that it can proceed with different tempo values for every step of the sequence in both the horizontal and vertical direction of the matrix sequencer. Figure 3.2 shows a basic principle behind a used model. Here every "f" object receives a corresponding value of length in ms and stores it. Trig. in #1 sends the value stored in the first "f" object to the metro object's second inlet (time argument) and thereby defines the time of the next trigger event.

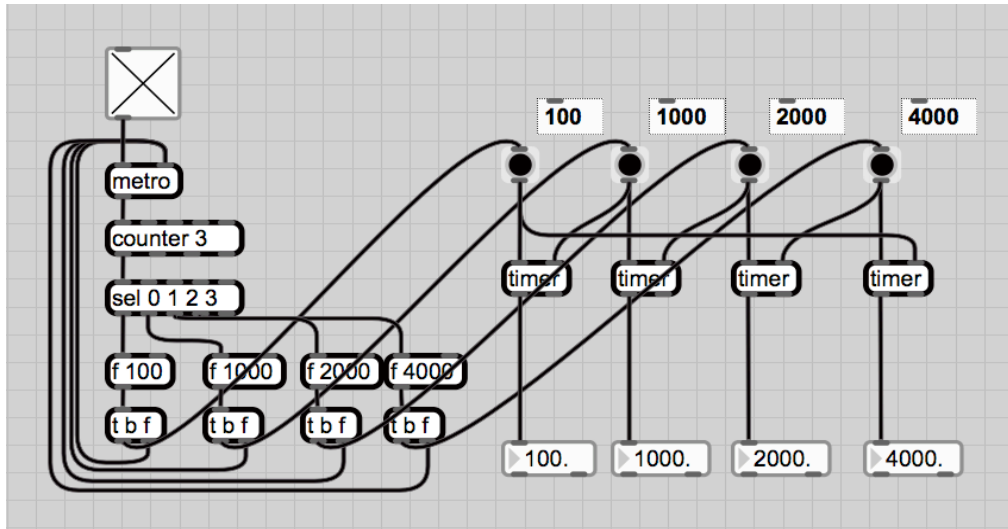


Figure 3.3 Example of the clock system

Every sequence line has its exclusive metro operator, where all the component of the row vector have to be sent. If all the sequencing lines are defined by different rule vectors, it is not possible to denote the output by using conventional music notation, because it is impossible to measure its durational determinants.

3.1.4 Amplitude envelopes and their main parameters

The two main objects in Max, which can be successfully used as amplitude envelope + amplifier systems, are the "function" and "adsr~". The "function" object is an essential and convenient tool for the implementation of arbitrary complex functions (trigonometric, piecewise and other) graphically. The main advantage here is that the user does not need to control various parameters of segments, because the envelope can be set in the function window by drawing. The output produced by the object is a linear, logarithmic or exponential ramp between n specified points (x,y) , where x is time in ms, and y is value at point x .

The crucial point in Max's "function" object is that this piecewise function is continuous, and if the domain of the attack is reduced by some constant c , it becomes $[0, x_a - c]$, the domain of the decay increases by the same constant and becomes $[x_a - c, x_d]$, the third point (sustain point) stays fixed; so by alteration of the attack segment we change the decay segment also. Simply put, the domain of the segment gets stretched if nearest segment is compressed. Therefore, controlling the sequencing with the function can be tedious on some

level, especially when the change of only one segment is needed. Another significant inconvenience comes, every time some arbitrary point enters a neighboring segment, it erases not only the border point, but the function of the segment itself.

The `adsr~` object allows the user to change a particular segment without intrusion into nearest segment. The `adsr~` uses a simple linear ramp and it does not compute any specific piecewise functions for the segments. The biggest disadvantage here, however, is that the ramp is always linear, and it can significantly reduce naturalness in fade-in. An ideal curve for fade-in should be some scale between logarithmic and linear, and one of the most popular choices is usually the quadratic curve (Puckette 2007, p. 92):

$$y[n] = (n/N)^4, \quad (3.1)$$

where N is a total number of samples to fade in, and n is a current sample.

In this project the author uses non-linear curves for all three fading segments: attack, decay and release. The envelope section, thus, programmed as more complex, is yet precise and an easy controlled envelope system.

When the clock system is activated and the domain vectors are defined properly, the block sends triggers to corresponding envelopes. The envelope block of every sequencing line consists of 16 independent envelopes, one personal envelope for each cell in the row.

Each row is programmed considering possible optimization problems, so it has one envelope engine and 16 data containers for each cell in the row. Therefore it is possible to define the ADSR parameters exclusively for every cell and store them as text in an appropriate data container, which in Max terminology is called a dictionary.

Parameters of the amplitude shape, which can be managed and stored for every element (individual cell) in a row are as follows: amplification level, attack, attack curve, decay, decay curve, sustain level, hold, release, release curve, and the scalars for attack, decay, hold and release parameters (Figure 3.4).

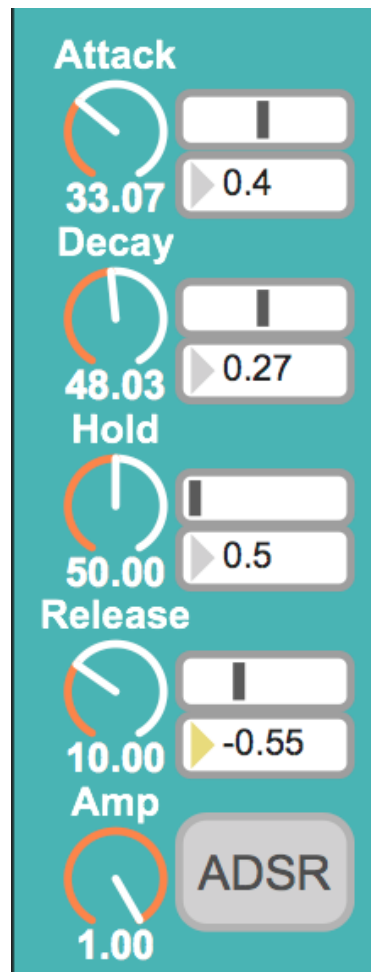


Figure 3.4 Amplitude envelope

Amplification level determines the maximum amplitude $f(x_a)$ in range $[0,1]$. This is a maximum value of amplitude, so during the attack segment the amplitude ramps from zero to $f(x_a)$ in x_a ms.

Attack segments are determined by a x_a , which is a position in time when fade in reaches the maximum and the slope of the curve changes the sign from positive to negative. **Curve of the attack** defines whether the attack's curve grows logarithmically or exponentially. In Max environment curve has a domain $[-1, 1]$, where the negative segment $[-1, 0)$ determines an exponential growth/decay, 0 stands for linear function and the segment $(0, 1]$ defines a logarithmic growth/decay of the curve. This feature is highly important especially for the rising segment, because human perceive amplitude changes in non-linear way. Hence the note magnitude shapes are never the same, it is a vital advantage to have the possibility of adjusting

and changing the curve in real time in order to produce variable shapes.

Decay determines the time of the fall from the maximum to the sustain level. Unlike the attack section curve, curve of the decay section has a negative slope, if $f(x_a) < f(x_d)$, decay becomes a continuation of the attack. In the given model, however, $f(x_d)$ is defined through a formula $c * f(x_a)$, where $c \in [0,1]$ is the sustain parameter, therefore $f(x_a) \geq f(x_d)$. **Curve of the decay**, similarly to the attack's curve, defines whether magnitude falls logarithmically or exponentially.

The slope of the decay falls to the **sustain** level, which is defined by the parameter $c \in [0,1]$. If $c = 1$, then the level of the sustain is equal to the length of the attack and the time of the sustain is extended, so it becomes $x_d + x_h$, where x_h is the time of the sustain. If $c = 0.5$ and attack is 0.5, then the actual sustain level will be $0.5 * 0.5 = 0.25$. The sustain determines the level of the flat magnitude (slop = 0), while the length of the sustain is managed by the **hold** position.

The **release** segment is the phase of the envelope signal where the envelope function has a negative slope and it falls from the sustain level to zero. Release is defined as a time function, and the release parameter manages the time of the fall. As the attack and decay, the release has a **release curve** parameter with the domain $[-1,1]$.

Each of these parameters has a fixed domain, but the range of Attack, Decay, Hold and Release can be modified by using of additional **scaling parameters**. The output ranges, which this scaling can provide are shown in Table 3.1

The maxim of the attack time for a given model of envelope will be 400 ms, if the scaling factor is set as 2. If, for instance, we use a hold range of 3, we can operate the hold parameter in the range of $[0,800]$ and so on.

Table 3.1 - Additional scaling of the Attack, Decay, Hold and Release outputs.

Scaling levels	0	1	2	3	4	5	6
Attack range	0-100	0-200	0-400	N/A	N/A	N/A	N/A
Decay range	0-100	0-200	0-400	N/A	N/A	N/A	N/A
Hold range	0-100	0-200	0-400	0-800	0-1600	N/A	N/A
Release range	0-100	0-200	0-400	0-800	0-1600	0-3200	0-6400

The last parameter in the envelope's model is an **ADSR/AD** switch. If the mode ADSR is selected, all four segments are used for the amplitude amplification. If AD mode is chosen, a Decay phase falls to zero and all following segments, Hold and Release, are ignored. AD mode is appropriate for a percussive sound objects processing.

3.1.5 Data routing inside the envelope row

The envelope row is the set of the amplitude envelope instances for every sequencing line. The data routing inside of the envelope, as it was mentioned before, is managed by using of special containers - dictionaries. Dictionaries receive, store and send appropriate data to the envelope engine. Data is sent when some particular dictionary is called. Figure 3.5 shows the part of this data routing with two dictionaries for one cell.

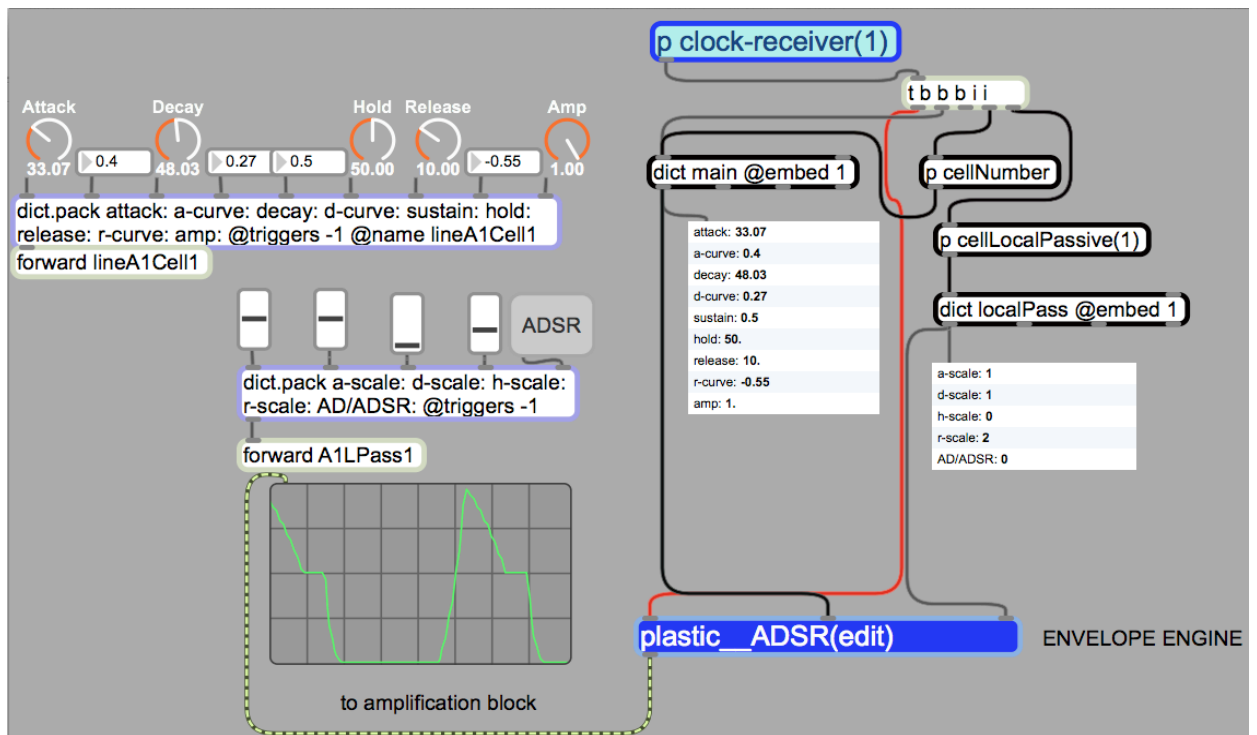


Figure 3.5 Data routing inside the envelope row

The first dictionary, named as "line1Cell1", packs a data for the attack, attack curve, decay, decay curve, sustain, hold, release, release curve and the level of the attack and sends the most recent statements (parameter name and its value) to the dictionary called "main" as soon as a corresponding cell is called. The second dictionary stores a data for scaling factors and ADSR/AD mode, labels the data as A1LPass1 and forwards it to the dictionary named "localPass" as soon as the corresponding cell gets the trigger message.

The object "p clock-receiver(1)" receives the bangs for the cells owned by the current sub-patch (in this example it is line #1 block), endows the bangs with a unique message (exclusive for every cell) and routes it to "cellNumber" and "cellLocalPassive(1)" sub-patches. When "cellNumber" receives a unique message, it replaces this message with the corresponding name of the dictionary (or the list), which the main dictionary has to receive (Figure 3.6). With this message the main dictionary receives the most recent data for all stored parameters. Similarly, "cellLocalPassive(1)" receives the same unique message, replaces it with corresponding name for "localPass" dictionary and sends the appropriate data to this dictionary (Figure 3.7).

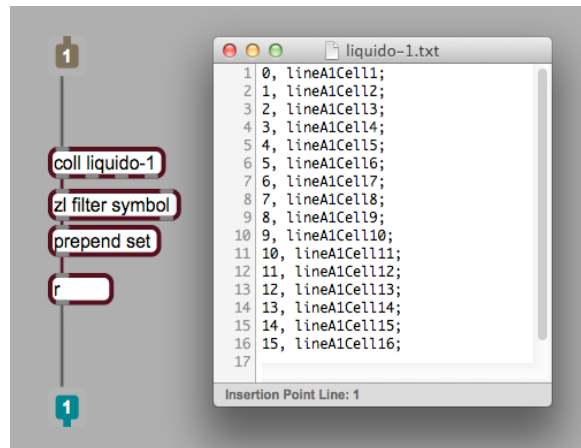


Figure 3.6 Data routing sub-patch for the dictionary "main"

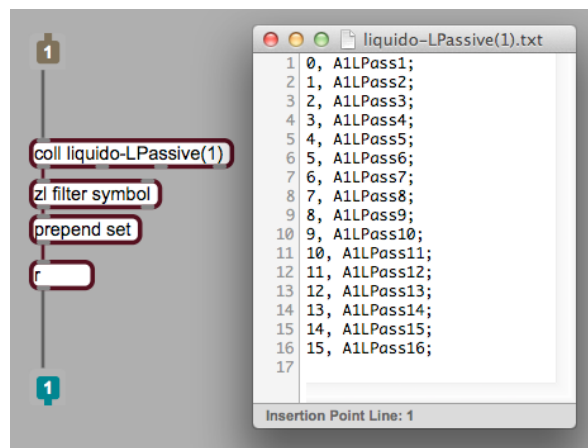


Figure 3.7 Data routing sub-patch for the dictionary "localPass"

After both dictionaries have received the data of the currently called sequencing cell, they send this data to the envelope engine, which replaces the old values of the envelope parameters with ones recently received. The signal from the envelope engine goes to the amplification block. The signal, produced by envelope configuration, which can be seen on the Figure 3.4, is depicted on Figure 3.6 with green curves and marked below with the note "to amplification block".

3.1.6 Matrix sequencer block

The matrix sequencer window consists of numeric data of vector rules (row vectors),

scaling factors and results of a vector multiplication by a scalar (scaling factors). This is the visual core of the sequencer, because here the artist defines a pattern for each row, modifies them by a scaling factor multiplication, synchronizes the sequences and loops them. The artist can modify selected rule vectors by changing their terms in the numeric boxes of the sequencer window (Figure 3.8). The scaled length value for every demential element in the matrix sequencer, which sent to the clock system, is stored here in white text boxes below the rule vector elements. The scaled matrix cannot be changed from these white boxes - their aim is to show a user the precise length for every step so he can modify corresponding amplitude envelopes.



Figure 3.8 Matrix Sequencer with 4 by 16 matrix size, consists of 4 sequencing rows and 16 step columns

In order to provide each cell and sequence row with extra possibilities modification of the patterns, looping mode is implemented used. **Looping** is useful when only part of the sequence is needed. The loops can be set in orange number boxes from the left side of the matrix sequencer. Figure 3.7 shows three rows with full 16-step length (1,2,4 rows) and one row with limited output [3,9]. The list starts with 0 index, so the range [3,9] means we have an actual loop from fourth to tenth element. The values of the minima and maxima for the rows can be set in orange number boxes shown on the right side of Figure 3.8.

3.1.7 Oscillators

For the demonstrating purpose, additional blocks of hybrid oscillators and filters are used in this work. A configuration of the hybrid oscillator based on additive synthesis (Roads 1997;

Puckette 2007; Russ ,) and complex frequency modulation (Chowing 1982). Oscillators are able to produce a wide range of harmonic and non-harmonic timbres.

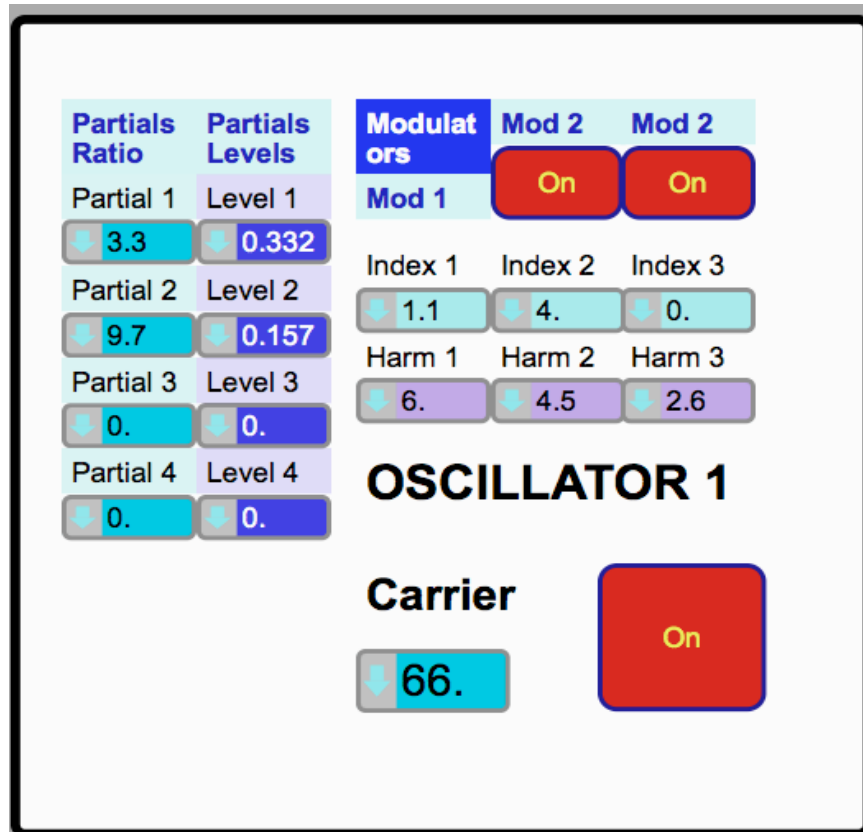


Figure 3.9 Oscillator

Each model consists of the following parameters: fundamental carrier; four partials with harmonicity factors (carrier multipliers) and gains; three modulators with harmonicity factors (ratio of carrier and modulating frequency) and modulation indices, which modulate the carrier and partials with the same frequencies (Figure ..). If harmonic sounds are needed, harmonicity factors for the partials and modulators should be integers.

3.2 Formative idea behind the model

The construction of abstract irrational sound patterns, which have a strong connection to neo-plasticism, is one of the main goals of the project. Considering the tenets of neo-plastic art, and a fundamental idea behind it, the author endowed the instrument with possibilities to process the atonal vertical structures, generate abstract patterns and route them through

monophonic outputs. As mentioned in Chapter 3, the author interprets the essential elements of Mondrian's paintings in the following way: every rectangle can be explained as a sound object; the shape of the sound object depicts geometrical properties of rectangle, where amplitude is a height and duration is a length; and color determines intensity of the sound object timbre, where white color corresponds to white noise, red depicted as red noise, blue - low frequency region, and yellow - high frequency region. Figure 3.10 depict the result accomplished with designed model, where the afore described interpretations are realized.

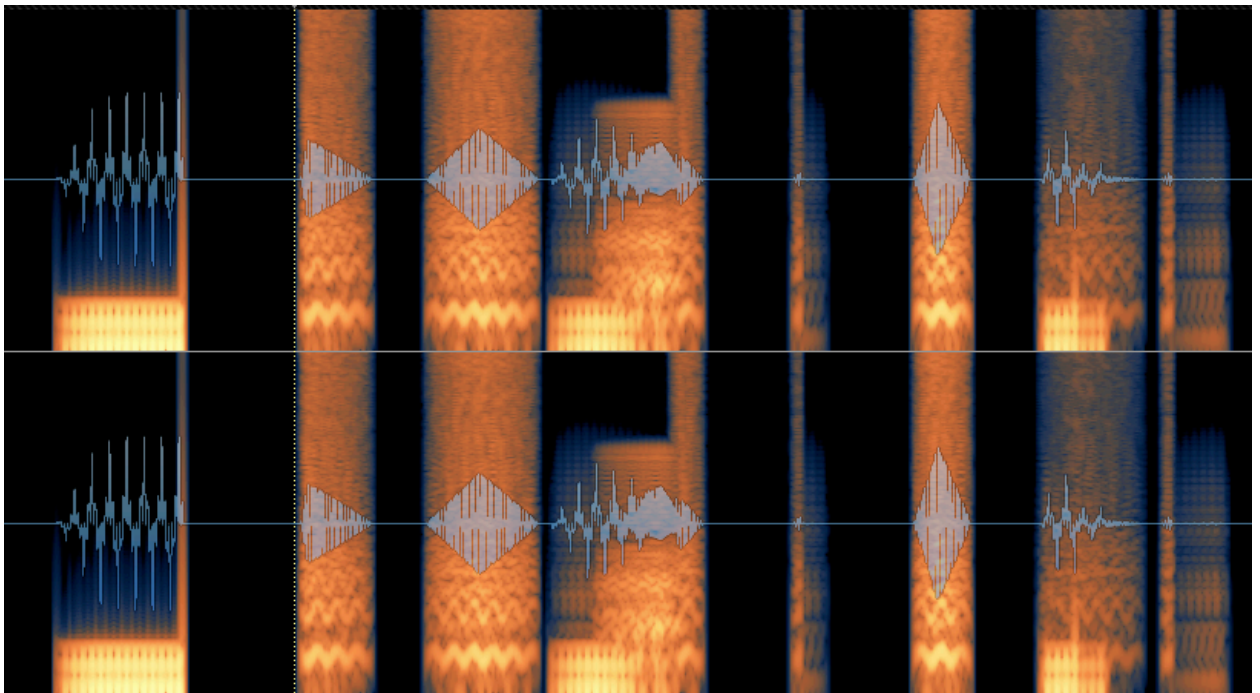


Figure 3.10 Spectrum and waveform view of the created pattern

3.3 Testing

Because the designed model has a complex data routing system, timing problems can occur. In order to prevent any desynchronization issues all duration data, which sent by clock engine, has to be received by envelopes without delay. Figures 3.11, 3.12, 3.13 and 3.14 show the tables of the test on data routing desynchronization results. The first column contains the values of the rule vector for every line, the second column - scaled values for clock system, the third column - measured time between bangs, which are sent from the clock system to the envelopes.

term1: 1.	line1step1: 103.	step1-1	103.000000
term2: 2.	line1step2: 206.	step1-2	206.000000
term3: 3.	line1step3: 309.	step1-3	309.000000
term4: 4.	line1step4: 412.	step1-4	412.000000
term5: 1.	line1step5: 103.	step1-5	103.000000
term6: 2.	line1step6: 206.	step1-6	206.000000
term7: 3.	line1step7: 309.	step1-7	309.000000
term8: 4.	line1step8: 412.	step1-8	412.000000
term9: 1.	line1step9: 103.	step1-9	103.000000
term10: 2.	line1step10: 206.	step1-10	206.000000
term11: 3.	line1step11: 309.	step1-11	309.000000
term12: 4.	line1step12: 412.	step1-12	412.000000
term13: 1.	line1step13: 103.	step1-13	103.000000
term14: 2.	line1step14: 206.	step1-14	206.000000
term15: 3.	line1step15: 309.	step1-15	309.000000
term16: 4.	line1step16: 412.	step1-16	412.000000

Figure 3.11 Line 1 test

term1: 44.	line3step1: 1144.	step1-1	153.000000
term2: 2.	line3step2: 52.	step1-2	17.000000
term3: 6.	line3step3: 156.	step1-3	34.000000
term4: 2.	line3step4: 52.	step1-4	102.000000
term5: 7.	line3step5: 182.	step1-5	17.000000
term6: 1.	line3step6: 26.	step1-6	102.000000
term7: 2.	line3step7: 52.	step1-7	51.000000
term8: 3.	line3step8: 78.	step1-8	153.000000
term9: 8.	line3step9: 208.	step1-9	17.000000
term10: 6.	line3step10: 156.	step1-10	119.000000
term11: 7.	line3step11: 182.	step1-11	136.000000
term12: 2.	line3step12: 52.	step1-12	153.000000
term13: 5.	line3step13: 130.	step1-13	34.000000
term14: 6.	line3step14: 156.	step1-14	102.000000
term15: 9.	line3step15: 234.	step1-15	102.000000
term16: 6.	line3step16: 156.	step1-16	68.000000

Figure 3.12 Line 2 test

term1: 44.	line3step1: 1144.	step1-1	1144.000000
term2: 2.	line3step2: 52.	step1-2	52.000000
term3: 6.	line3step3: 156.	step1-3	156.000000
term4: 2.	line3step4: 52.	step1-4	52.000000
term5: 7.	line3step5: 182.	step1-5	182.000000
term6: 1.	line3step6: 26.	step1-6	26.000000
term7: 2.	line3step7: 52.	step1-7	77.000000
term8: 3.	line3step8: 78.	step1-8	78.000000
term9: 8.	line3step9: 208.	step1-9	208.000000
term10: 6.	line3step10: 156.	step1-10	156.000000
term11: 7.	line3step11: 182.	step1-11	208.000000
term12: 2.	line3step12: 52.	step1-12	52.000000
term13: 5.	line3step13: 130.	step1-13	130.000000
term14: 6.	line3step14: 156.	step1-14	156.000000
term15: 9.	line3step15: 234.	step1-15	259.000000
term16: 6.	line3step16: 156.	step1-16	156.000000

Figure 3.13 Line 3 test

term1: 1.	line4step1: 126.	step1-1	126.000000
term2: 1.	line4step2: 126.	step1-2	126.000000
term3: 1.	line4step3: 126.	step1-3	126.000000
term4: 1.	line4step4: 126.	step1-4	195.000000
term5: 1.	line4step5: 126.	step1-5	126.000000
term6: 1.	line4step6: 126.	step1-6	126.000000
term7: 1.	line4step7: 126.	step1-7	126.000000
term8: 1.	line4step8: 126.	step1-8	126.000000
term9: 1.	line4step9: 126.	step1-9	126.000000
term10: 1.	line4step10: 126.	step1-10	126.000000
term11: 1.	line4step11: 126.	step1-11	126.000000
term12: 1.	line4step12: 126.	step1-12	126.000000
term13: 1.	line4step13: 126.	step1-13	126.000000
term14: 1.	line4step14: 126.	step1-14	126.000000
term15: 1.	line4step15: 126.	step1-15	126.000000
term16: 1.	line4step16: 126.	step1-16	126.000000

Figure 3.14 Line 4 test

All the test have proved that the clock system proceeds data synchronously.

CONCLUSION

As it was stated in the introduction, the aim of the current work was to design a sequencer model that could serve as an initial template for an artist. With such a template an artist could construct, modify and adjust his own abstract patterns in real time. The instrument aimed to provide an artist with a flexible workflow for a live performance and artistic improvisation. The method of construction was built on mathematical grounding on the one hand, and expressive creativity on the other.

Evaluating the results, it may be concluded that the matrix sequencer with variable domain is suitable for electronic music improvisation and can be used as a real-time instrument. It provides an artist with possibilities to develop his own configuration of the sequencer. At the same time, it is important to mention that this sequencer model cannot substitute a complete instrument, because it was designed initially as a template, the domain of which can be modified in various ways – by a rule, manually, or randomly. One of the biggest advantages of this model is that all four channels or rows of the matrix can be defined by different rules. Taking as a basis the analogue sequencer concepts, the author concludes that this model expands possibilities of the modular systems and makes the same operations more precise.

Yet there is a room for further research, especially in the direction of optimization of data routing and performance. At this stage of the project, the initial ideas were implemented with positive results, and therefore the project could be expanded and optimized further. Thus, the control system should be improved and simplified, so that an artist will not spend much time for the adjustment of the template. Another direction to work with is the optimization of the data routine. It can be done by replacing the cumbersome structure of amplitude envelope sub-patches through coding.

Due to intense data flow receiving, the clock system must be optimized as well. For instance, when a constant tempo is used it must have an option to switch the mode for system optimization. Testing on data routing showed that many of the passive wires and objects are used when it does not needed. Certainly, it can affect stability of the template. Taking into account Roads' remark that amplitude shaping is a delicate operation (Roads 1997), it is necessary to modify the amplitude envelope blocks by adding complex graphical shapes, while at the same time preserving simple control.

Another crucial improvement can be made by adjusting the signal processing methods of masking and shift-and-ramp for discontinuity fixing (Puckette 2007, pp. 95-96). Alternatively, the psychoacoustical method of masking one sound by another can be used in order to mask the audible artifacts (Howard & Angus 2009, pp. 260–261).

The possible future research and modification can be done in the direction of interface modernization using an artistic approach. Massive ADSR lines and oscillators can be replaced with geometrical figures (rectangles), where the color and shade of the rectangle will be engaged with frequency and modulation parameters of the oscillator, width with the duration of the sound object and height with its magnitude.

REFERENCES

- Beech, D.** 2009. *Documents of Contemporary Art*. Cambridge: The MIT Press.
- Benson, J.** 2007. *Music.A Mathematical Offering*. Cambridge:Cambridge University Press.
- Bilbao, S.** 2009. Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics. Chichester: John Wiley & Sons.
- Bradley, S.P., Hax, A.C., Magnanti T.L.** 1997. Applied Mathematical Programming. Boston: Addison Wesley.
- Burrows,** 1999. *How to read music*. New York :Carlton Books Limited.
- Chowing, J., Bristow D.** 1986. *FM Theory & Applications. By Musicians for Musicians*. Tokyo: Yamaha Music Foundation.
- Cipriani, A., Giri, M.** 2009. *Electronic Music and Sound Design. Theory and Practice with Max/MSP - Vol.1*.Rome: ConTempoNet s.a.s.
- Deleuze, G.** 2003. *Francis Bacon: The Logic of Sensation*, New-York, London. Continuum
<http://www-ccrma.stanford.edu/~jos/mdft/> (21.03.2014).
- Doesburg, T.** 1919. *Principles Of Neo-plastic Art*. In Art and Theory 1900-1919. An Anthology of Changing Ideas. Oxford: Blackwell Publishers Ltd.
- Howard, D., Angus, J.** 2009. Acoustics and Psychoacoustics. Fourth Edition. Burlington: Elsevier Ltd.

Ingle, V., Proakis, J. 2012. *Digital Signal Processing using MATLAB*. Third edition. Stamford: Cengage Learning.

Leis, J. 2011. *Digital Signal Processing Using MATLAB For Students and Researchers*. New Jersey: John Wiley & Sons.

Massie, D.C. 1997. *Wavetable Sampling Synthesis*. In *Applications of digital signal processing to audio and acoustics*. Boston, Dordrecht, London: Kluwer Academic publishers.

Miranda, E. 2002. *Computer Sound Design*. Second Edition. Oxford: Focal Press.

Mondrian, P. 1919. *Dialogue on the New Plastic*. In *Art and Theory 1900-1919. An Anthology of Changing Ideas*. Oxford: Blackwell Publishers Ltd.

Prandoni, P. and Vetterli, M. 2008. *Signal Processing for Commutators*. Lausanne: EPFL Press.

Puckette, M. 2006. *The Theory and Technique of Electronic Music*. World Scientific Publishing Co. Pte. Ltd.

Roads, C. 1996. *The Computer Music Tutorial*. Cambridge: MIT Press.

Russ, M. 2009. *Sound Synthesis and Sampling*. Third Edition. Oxford: Elsevier Ltd.
Sequences and series. 2013. Department of Mathematics, University of the Aegean.

Smith, J.O. 2002. *Mathematics of the Discrete Fourier Transform*. Stanford: Center for Computer Research in Music and Acoustics (CCRMA).

Smith, J.O. 2007. *Introduction to Digital Filters with Audio Applications*.
<http://ccrma.stanford.edu/~jos/filters/>, online book (29.01.2014).

Strang, G. 1991. *Calculus Online Textbook*. Cambridge: Wellesley-Cambridge Press.

Thomas, G.B., Maurice D.W. 2004. *Thomas' Calculus*. Boston: Addison Wesley Longman.

Wickert, M. 2011. *Introduction to signals and systems*. [Lecture notes].

<http://www.eas.uccs.edu/wickert/ece2610/> (02.02.2014).

Wright, D. 2009. *Mathematics and Music*. American Mathematical Society.

SUMMARY

The 20th century was the century of the music production, while the 21st is the century of the live performances and installations. Electronic music is generally associated with the technological progress. Technical innovations in designing software instruments have been playing a crucial role in the development of electronic music. The most progressive electronic music artists start their work at the stage of the designing of the technical tools for their projects, therefore electronic music has become a conceptual bridge between music, art, architecture and science. The matrix sequencer with variable domain aimed to provide an artist with a flexible workflow for a live performance and artistic improvisation. The method of construction was built on mathematical grounding on the one hand, and expressive creativity on the other.